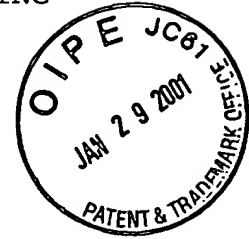


SUBSTITUTE SPECIFICATIONTUNABLE NARROW-BAND FILTER INCLUDING
SIGMA-DELTA MODULATORChristopher H. Dick
Frederic J. Harris**CROSS-REFERENCE TO RELATED APPLICATION**

This is a continuation-in-part of U.S. Patent Application Serial No. 09/394,123 filed 9/10/1999 entitled "Narrow Band Filter Including Sigma-Delta Modulator Implemented in a Programmable Logic Device", incorporated herein by reference.

BACKGROUND

While $\Sigma\Delta$ techniques are applied widely in analog conversion sub-systems, both analog-to-digital (ADC) and digital-to-analog (DAC) converters, these methods have enjoyed much less exposure in the broader application domain, where flexible and configurable solutions, traditionally supplied via a software DSP (soft-DSP), are required. And this limited level of exposure is easy to understand. Most, if not all, of the efficiencies and optimizations afforded by $\Sigma\Delta$ are hardware oriented and so cannot be capitalized on in the fixed precision pre-defined datapath found in a soft-DSP processor. This limitation, of course, does not exist in a field programmable gate array (FPGA) DSP solution. With FPGAs the designer has complete control of the silicon to implement any desired datapath and employ optimal word precisions in the system with the objective of producing a design that satisfies the specifications in the most economically sensitive manner.

While implementation of a digital $\Sigma\Delta$ ASIC (application-specific integrated circuit) is of course possible, economic

SUBSTITUTE SPECIFICATION

1 constraints make the implementation of such a building block
2 that would provide the flexibility, and be generic enough to
3 cover a broad market cross-section, impractical. FPGA-based
4 hardware provides a solution to this problem. FPGAs are off-
5 the-shelf commodity items that provide a silicon feature set
6 ideal for constructing high-performance DSP systems. These
7 devices maintain the flexibility of software-based
8 solutions, while providing levels of performance that match,
9 and often exceed, ASIC solutions.

10 There is a rich and expanding body of literature
11 devoted to the efficient and effective implementation of
12 digital signal processors using FPGA based hardware. More
13 often than not, the most successful of these techniques
14 involves a paradigm shift away from the methods that provide
15 good solutions in software programmable DSP systems.

16 Semiconductor vendors, such as Xilinx, Altera, Atmel,
17 and AT&T, provide a range of FPGAs. The architectural
18 approaches are as diverse as there are manufacturers, but
19 some generalizations can be made. Most of the devices are
20 basically organized as an array of logic elements and
21 programmable routing resources used to provide the
22 connectivity between the logic elements, FPGA I/O pins and
23 other resources, such as on-chip memory. The structure and
24 complexity of the logic elements, as well as the
25 organization and functionality supported by the
26 interconnection hierarchy, distinguish the devices. Other
27 device features, such as block memory and delay locked loop
28 technology, are also significant factors that influence the
29 complexity and performance of an algorithm that is
30 implemented using FPGAs.

31 A logic element usually consists of one or more RAM
32 (random access memory) n-input look-up tables, where n is

SUBSTITUTE SPECIFICATION

1 between 3 and 6, in one to several flip-flops. There may
2 also be additional hardware support in each element to
3 enable high-speed arithmetic operations. This generic FPGA
4 architecture is shown in Figure 1. Also illustrated in the
5 Figure (as wide lines) are several connections between logic
6 elements and the device input/output (I/O) ports.

7 Application-specific circuitry is supported in the device by
8 downloading a bit stream into SRAM (static random access
9 memory) based configuration memory. This personalization
10 database defines the functionality of the logic elements, as
11 well as the internal routing. Different applications are
12 supported on the same FPGA hardware platform by configuring
13 the FPGA(s) with appropriate bit streams. As a specific
14 example, consider the Xilinx Virtex™ series of FPGAs. The
15 logic elements, called slices, essentially consist of two
16 four-input look-up tables (LUTs), two flip-flops, several
17 multiplexors and some additional silicon support that allows
18 the efficient implementation of carry-chains for building
19 high-speed adders, subtracters, and shift registers. Two
20 slices form a configurable logic block (CLB) as shown in
21 Figure 2. The CLB is the basic tile that is used to build
22 the logic matrix. Some FPGAs, like the Xilinx Virtex
23 families, supplying on-chip block RAM. Figure 3 shows the
24 CLB matrix that defines a Virtex FPGA. Current generation
25 Virtex silicon provides a family of devices offering 768 to
26 12,288 logic slices, and from 8 to 32 variable form factor
27 block memories.

28 Xilinx XC4000 and Virtex devices also allow the
29 designer to use the logic element LUTs as memory, either ROM
30 or RAM. Constructing memory with this distributed memory
31 approach can yield access bandwidths in many tens of
32 gigabytes per second range.

SUBSTITUTE SPECIFICATION

1 Typical clock frequencies for current generation
2 devices are in the multiple tenants of megahertz (100 to
3 200) range.

4 In contrast to the logic slice architecture employed in
5 Xilinx Virtex devices, a logic block architecture employed
6 in the Atmel AT40K FPGA is shown in Figure 4. Like the
7 Xilinx device, combinational logic is realized using look-up
8 tables. In this case, two three-input LUTs and a single
9 flip-flop are available in each logic cell. The pass gates
10 in a cell form part of the signal routing network and are
11 used for connecting signals to the multiple horizontal and
12 vertical bus planes. In addition to the orthogonal routing
13 resources, indicated as N, S, E and W in Figure 4, a
14 diagonal group of interconnects (NW, NE, SE, and SW),
15 associated with each cell x output, are available to provide
16 efficient connections to neighboring cell's x bus inputs.

17 The objective of the FPGA/DSP architect is to formulate
18 algorithmic solutions for applications that best utilize
19 FPGA resources to achieve the required functionality. This
20 is a three-dimensional optimization problem in power,
21 complexity, and bandwidth. The remainder of this application
22 describes some novel FPGA solutions to several signal
23 processing problems. The results are important in an
24 industrial context because they enable either smaller, and
25 hence more economic, solutions to important problems, or
26 allow more arithmetic compute power to be realized with a
27 given area of silicon.

28

29 $\Sigma\Delta$ Modulation

30 Sigma-Delta modulation is a source coding technique
31 most prominently employed in analog-to-digital and digital-

SUBSTITUTE SPECIFICATION

1 to-analog converters. In this context, hybrid analog and
 2 digital circuits are used in the realization. Figure 5 shows
 3 a single-loop $\Sigma\Delta$ modulator. Provided the input signal is
 4 busy enough, the linearized discrete time model of Figure 6
 5 can be used to illustrate the principle. In Figure 6, the 1-
 6 bit quantizer is modeled by an additive white noise source
 7 with variance $\sigma_e^2 = \Delta^2/12$, where Δ represents the quantization
 8 interval. The z-transform of the system is

9

10 Equations 1 and 2:

$$\begin{aligned} Y(z) &= \frac{H(z)}{1 + H(z)} X(z) + \frac{1}{1 + H(z)} Q(z) \\ &= H_s(z) X(z) + H_n(z) Q(z) \end{aligned}$$

11 where

12

13 Equation 3:

$$H(z) = \frac{1}{z - 1}$$

14

15 which is the transfer function of delay and an ideal
 16 integrator, and $H_s(z)$ and $H_n(z)$ are the signal and noise
 17 transfer functions (NTF) respectively. In a good $\Sigma\Delta$
 18 modulator, $H_n(\omega)$ will have a flat frequency response in the
 19 interval $|f| \leq B$. In contrast, $H_n(\omega)$ will have a high
 20 attenuation in the frequency band $|f| \leq B$ and a "don't care"
 21 region in the interval $B < |f| < f_s/2$. For the single loop $\Sigma\Delta$
 22 in Figure 6, $H_s(z) = z^{-1}$ and $H_n(z) = 1 - z^{-1}$. Thus the input
 23 signal is not distorted in any way by the network and simply
 24 experiences a pure delay from input to output. The
 25 performance of the system is determined by the noise
 26 transfer function $H_n(z)$, which is given by

SUBSTITUTE SPECIFICATION

1

2 Equation 4:

$$|H_n(f)| = 4 \left| \sin \frac{\pi f}{f_s} \right|$$

3

4 and is shown in Figure 7. The in-band quantization noise
5 variance is

6

7 Equation 5:

$$\sigma_n^2 = \int_{-B}^{+B} |H_n(f)|^2 S_q(f) df$$

8

9 where $S_q(f) = \sigma_q^2 / f_s$ is the power spectral density of the
10 quantization noise. Observe that for a non-shaped noise (or
11 white) spectrum, increasing the sampling rate by a factor of
12 2, while keeping the bandwidth B fixed, reduces the
13 quantization noise by 3 dB. For a first order $\Sigma\Delta$ M it can be
14 shown that

15

16 Equation 6:

$$\sigma_n^2 \approx \frac{1}{3} \pi^2 \sigma_q^2 \left(\frac{2B}{f_s} \right)^3$$

17

18 for $f_s \gg 2B$. Under these conditions doubling the sampling
19 frequency reduces the noise power by 9 dB, of which 3 dB is
20 due to the reduction in $S_q(f)$ and a further 6 dB is due to
21 the filter characteristic $H_n(f)$. The noise power is reduced
22 by increasing the sampling rate to spread the quantization
23 noise over a large bandwidth and then by shaping the power
24 spectrum using an appropriate filter.

SUBSTITUTE SPECIFICATIONReduced Complexity Filters Using $\Sigma\Delta$ Modulation Techniques

$\Sigma\Delta$ techniques can be employed for realizing area efficient narrowband filters in FPGAs. These filters are utilized in many applications. For example, narrow-band communication receivers, multi-channel RF surveillance systems and for solving some spectrum management problems.

A uniform quantizer operating at the Nyquist rate is the standard solution to the problem of representing data within a specified dynamic range. Each additional bit of resolution in the quantizer provides an increase in dynamic range of approximately 6dB. A signal with 60dB of dynamic range requires 10 bits, while 16 bits can represent data with a dynamic range of 96dB.

While the required dynamic range of a system fixes the number of bits required to represent the data, it also affects the expense of subsequent arithmetic operations, in particular multiplications. In any hardware implementation, and of course this includes FPGA based DSP processors, there are strong economic imperatives to minimize the number and complexity of the arithmetic components employed in the datapath. An embodiment of the invention employs noise-shaping techniques to reduce the precision of the input data samples to minimize the complexity of the multiply-accumulate (MAC) units in the filter. The net result is a reduction in the amount of FPGA logic resources required to realize the specified filter.

Consider the structure shown in Figure 8. Instead of applying the quantized data $x(n)$ from the analog-to-digital converter directly to the filter, data $x(n)$ is pre-processed by a $\Sigma\Delta$ modulator. The re-quantized input samples $\hat{x}(n)$ are

SUBSTITUTE SPECIFICATION

1 represented using fewer bits per sample, so permitting the
2 subsequent filter $H(z)$ to employ reduced precision
3 multipliers in the mechanization. The filter coefficients
4 are still kept to a high precision.

5 The $\Sigma\Delta$ data re-quantizer is based on a single loop
6 error feedback sigma-delta modulator shown in Figure 9. In
7 this configuration, the difference between the quantizer
8 input and output sample is a measure of the quantization
9 error, which is fed back and combined with the next input
10 sample. The error-feedback sigma-delta modulator operates on
11 a highly oversampled input and uses the unit delay z^{-1} as a
12 predictor. With this basic error-feedback modulator, only a
13 small fraction of the bandwidth can be occupied by the
14 required signal. In addition, the circuit only operates at
15 baseband. A larger fraction of the Nyquist bandwidth can be
16 made available and the modulator can be tuned if a more
17 sophisticated error predictor is employed. This requires
18 replacing the unit delay with a prediction filter $P(z)$. This
19 generalized modulator is shown in Figure 10.

20 The operation of the re-quantizer can be understood by
21 considering the transform domain description of the circuit.
22 This is expressed as

23

24 Equation 7:

$$\hat{X}(z) = X(z) + Q(z)(1 - P(z)z^{-1})$$

25

26 where $Q(z)$ is the z -transform of the equivalent noise source
27 added by the quantizer $q(\cdot)$, $P(z)$ is the transfer function of
28 the error predictor filter, and $X(z)$ and $\hat{X}(z)$ are the
29 transforms of the system input and output respectively. $P(z)$
30 is designed to have unity gain and leading phase shift in

SUBSTITUTE SPECIFICATION

1 the bandwidth of interest. Within the design bandwidth, the
2 term $Q(z)(1-P(z)z^{-1})=0$ and so $X(z) = \hat{X}(z)$. By designing $P(z)$
3 to be commensurate with the system passband specifications,
4 the in-band spectrum of the re-quantizer output will ideally
5 be the same as the corresponding spectral region of the
6 input signal.

7 To illustrate the operation of the system consider the
8 task of recovering a signal that occupies 10% of the
9 available bandwidth and is centered at a normalized
10 frequency of 0.3Hz. The stopband requirement is to provide
11 60 dB of attenuation. Figure 11A shows the input test
12 signal. It comprises an in-band component and two out-of-
13 band tones that are to be rejected. Figure 11B is a
14 frequency domain plot of the signal after it has been re-
15 quantized to 4 bits of precision by a $\Sigma\Delta$ modulator employing
16 an 8th order predictor in the feedback path. Notice that the
17 60dB dynamic range requirement is supported in the bandwidth
18 of interest, but that the out-of-band SNR has been
19 compromised. This is of course acceptable, since the
20 subsequent filtering operation will provide the necessary
21 rejection. A 160-tap filter $H(z)$ satisfies the problem
22 specifications. The frequency response of $H(z)$ using 12-bit
23 filter coefficients is shown in Figure 11C. Finally, $H(z)$ is
24 applied to the reduced sample precision data stream $\hat{X}(z)$ to
25 produce the spectrum shown in Figure 11D. Observe that the
26 desired tone has been recovered, the two out-of-band
27 components have been rejected, and that the in-band dynamic
28 range meets the 60 dB requirement.

29

SUBSTITUTE SPECIFICATIONPrediction Filter Design

The design of the error predictor filter is a signal estimation problem. The optimum predictor is designed from a statistical viewpoint. The optimization criterion is based on the minimization of the mean-squared error. As a consequence, only the second-order statistics (autocorrelation function) of a stationary process are required in the determination of the filter. The error predictor filter is designed to predict samples of a band-limited white noise process $N_{xx}(\omega)$ shown in Figure 12.

$N_{xx}(\omega)$ is defined as:

Equation 8:

$$N_{xx}(\omega) = \begin{cases} 1 & -\theta \leq \omega \leq \theta \\ 0 & \text{otherwise} \end{cases}$$

and related to the autocorrelation sequence $r_{xx}(m)$ by discrete-time Fourier transform (DTFT).

Equation 9:

$$N_{xx}(\omega) = \sum_{n=-\infty}^{\infty} r_{xx}(n) e^{-j\omega n}$$

The autocorrelation function $r_{xx}(n)$ is found by taking the inverse DTFT of the equation immediately above.

Equation 10:

$$r_{xx}(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} N_{xx}(\omega) e^{j\omega n} d\omega$$

$N_{xx}(\omega)$ is non-zero only in the interval $-\theta \leq \omega \leq \theta$ giving $r_{xx}(n)$ as:

SUBSTITUTE SPECIFICATION

1

2 Equation 11:

3

$$r_{xx}(n) = \frac{\theta}{\pi} \text{sinc}(\theta n)$$

4

5 So the autocorrelation function corresponding to a band-
 6 limited white noise power spectrum is a sinc function. Samples
 7 of this function are used to construct an autocorrelation
 8 matrix which is used in the solution of the normal equations
 9 to find the required coefficients. Leaving out the scaling
 10 factor in the immediately above equation, the required
 11 autocorrelation function $r_{xx}(n)$, truncated to p samples, is
 12 defined as:

13

14 Equation 12:

15

16

$$r_{xx} = \frac{\sin(n\theta)}{n\theta} \quad n = 0, \dots, p-1$$

17

18

19 The normal equations are defined as:

20

21 Equation 13:

22

23

$$r_{xx}(m) = \sum_{k=1}^p a(k) r_{xx}(m-k) \quad m = 1, 2, \dots, p$$

24

25

26 This system of equations can be compactly written in
 27 matrix form by first defining several matrices.

28 To design a p-tap error predictor filter first compute a
 29 sinc function consisting of p+1 samples and construct the
 30 autocorrelation matrix R_{xx} as:

31

SUBSTITUTE SPECIFICATION

Equation 14:

$$R_{xx} = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \dots & r_{xx}(p-1) \\ r_{xx}(1) & r_{xx}(0) & \dots & r_{xx}(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}(p-1) & r_{xx}(p-2) & \dots & r_{xx}(0) \end{bmatrix}$$

Next, define a filter coefficient row-vector A as:

Equation 15:

$$A = [a(0), a(1), \dots, a(p-1)]$$

where $a(i), i=0, \dots, p-1$, are the predictor filter coefficients. Let the row-vector R'_{xx} be defined as:

Equation 16:

$$R'_{xx} = [r_{xx}(1), r_{xx}(2), \dots, r_{xx}(p)]$$

The matrix equivalent of equation 13 is:

Equation 17:

$$R_{xx} A^T = (R'_{xx})^T$$

The filter coefficients are therefore given as:

Equation 18:

$$A^T = R_{xx}^{-1} (R'_{xx})^T$$

SUBSTITUTE SPECIFICATION

For the case in-hand, the solution of equation 18 is an ill-conditioned problem. To arrive at a solution for A, a small constant ϵ is added to the elements along the diagonal of the autocorrelation matrix R_{xx} in order to raise its condition number. The actual autocorrelation matrix used to solve for the predictor filter coefficients is:

Equation 19:

$$R_{xx} = \begin{bmatrix} r_{xx}(0)+\epsilon & r_{xx}(1) & \dots & r_{xx}(p-1) \\ r_{xx}(1) & r_{xx}(0)+\epsilon & \dots & r_{xx}(M-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}(p-1) & r_{xx}(p-2) & \dots & r_{xx}(0)+\epsilon \end{bmatrix}$$

Bandpass Predictor Filter

The previous section described the design of a lowpass predictor. In this section, bandpass processes are considered.

A bandpass predictor filter is designed by modulating a lowpass prototype sinc function to the required center frequency θ_0 . The bandpass predictor coefficient $h_{BP}(n)$ is obtained from the prototype lowpass sinc function $h_{LP}(n)$ as:

Equation 20:

$$\text{sinc}_{BP}(n) = \text{sinc}_{LP}(n) \cos(\theta_0(n-k)) \quad n = 0, \dots, 2p$$

$$\text{where } k = \left\lceil \frac{2p+1}{2} \right\rceil.$$

SUBSTITUTE SPECIFICATION1 Highpass Predictor Filter

2 A highpass predictor filter is designed by highpass
 3 modulating a lowpass prototype sinc function to the required
 4 corner frequency θ_c . The highpass predictor coefficients $h_{HP}(n)$
 5 are obtained from the prototype lowpass sinc function $h_{LP}(n)$
 6 as:

7

8 Equation 21:

9

$$\text{sinc}_{HP}(n) = \text{sinc}_{LP}(n) (-1)^{n-k} \quad n = 0, \dots, 2p$$

10

11

12 $\Sigma\Delta$ Modulator FPGA Implementation

13 The most challenging aspect of implementing the data
 14 modulator is producing an efficient implementation for the
 15 prediction filter $P(z)$. The desire to support high-sample
 16 rates, and the requirement of zero latency for $P(z)$, will
 17 preclude bit-serial methods from this problem. In addition,
 18 for the sake of area efficiency, parallel multipliers that
 19 exploit one time-invariant input operand (the filter
 20 coefficients) will be used, rather than general variable-
 21 variable multipliers. The constant-coefficient multiplier
 22 (KCM) is based on a multi-bit inspection version of Booth's
 23 algorithm. Partitioning the input variable into 4-bit
 24 nibbles is a convenient selection for the Xilinx Virtex
 25 function generators (FG). Each FG has 4 inputs and can be
 26 used for combinatorial logic or as application RAM/ROM. Each
 27 logic slice in the Virtex logic fabric comprises 2 FGs, and
 28 so can accommodate a 16×2 memory slice. Using the rule of
 29 thumb that each bit of filter coefficient precision
 30 contributes 5 dB to the sidelobe behavior, 12-bit precision
 31 is used for $P(z)$. 12-bit precision will also be employed for

SUBSTITUTE SPECIFICATION

1 the input samples. There are 3 4-bit nibbles in each input
2 sample. Concurrently, each nibble addresses independent $16 \times$
3 16 lookup tables (LUTs). The bit growth incorporated here
4 allows for worst case filter coefficient scaling in $P(z)$. No
5 pipeline stages are permitted in the multipliers because of
6 $P(z)$'s location in the feedback path of the modulator.

7 It is convenient to use the transposed FIR filter for
8 constructing the predictor. This allows the adders and delay
9 elements in the structure to occupy a single slice. 64
10 slices are required to build the accumulate-delay path. The
11 FPGA logic requirements for $P(z)$, using a 9-tap predictor,
12 is $\Gamma(P(z)) = 9 \times 40 + 64 = 424$ CLBs. A small amount of
13 additional logic is required to complete the entire $\Sigma\Delta$
14 modulator. The final slice count is 450. The entire
15 modulator comfortably operates with a 113 MHz clock. This
16 clock frequency defines the system sample rate, so the
17 architecture can support a throughput of 113 MSamples per
18 second. The critical path through this part of the design is
19 related to the exclusion of pipelining in the multipliers.

20

21 Reduced Complexity FIR Mechanization

22 Now that the input signal is available as a reduced
23 precision sample stream, filtering can be performed using
24 area-optimized hardware. For the reasons discussed above, 4-
25 bit data samples are a convenient match for Virtex devices.
26 Figure 13 shows the structure of the reduced complexity FIR
27 filter. The coded samples $\hat{x}(n)$ are presented to the address
28 inputs of N coefficient LUTs. In accordance with the
29 modulated data stream precision, each LUT stores the 16
30 possible scaled coefficient values for one tap as shown in
31 Figure 14. An N -tap filter requires N such elements. The

SUBSTITUTE SPECIFICATION

1 outputs of the minimized multipliers are combined with an
2 add-delay datapath to produce the final result. The logic
3 requirement for the filter is $\Gamma(H(z)) = N\Gamma(MUL) + (N-1)\Gamma(ADD_z^{-1})$
4 where $\Gamma(MUL)$ and $\Gamma(ADD_z^{-1})$ are the FPGA area cost functions
5 for a KCM multiplier and an add-delay datapath component
6 respectively.

7 Using full-precision input samples without any $\Sigma\Delta$
8 encoding, each KCM would occupy 40 slices. The total cost of
9 a direct implementation of $H(z)$ is 7672 slices. The reduced
10 precision KCMs used to process the encoded data each consume
11 only 8 slices. Including the sigma-delta modulator the slice
12 count is 3002 for the $\Sigma\Delta$ approach. So the data re-
13 quantization approach consumes only 39% of the logic
14 resources of a direct implementation.

15

16 $\Sigma\Delta$ Decimators,

17 The procedure for re-quantizing the source data can
18 also be used effectively in an m:1 decimation filter. An
19 interesting problem is presented when high input sample
20 rates (≥ 150 MHz) must be supported in FPGA technology. High-
21 performance multipliers are typically realized by
22 incorporating pipelining in the design. This naturally
23 introduces some latency in to the system. The location of
24 the predictor filter $P(z)$ requires a zero-latency design.
25 (It is possible that the predictor could be modified to
26 predict samples further ahead in the time series, but this
27 potential modification will not be dealt with in the limited
28 space available.) Instead of re-quantizing, filtering and
29 decimating, which would of course require a $\Sigma\Delta$ modulator
30 running at the input sample rate, this sequence of
31 operations is re-ordered to permit several slower modulators

SUBSTITUTE SPECIFICATION

1 to be used in parallel. The process is performed by first
2 decimating the signal, re-quantizing and then filtering. Now
3 the $\Sigma\Delta$ modulators operate at the reduced output sample rate.
4 This is depicted in Figure 15. To support arbitrary center
5 frequencies, and any arbitrary, but integer, down-sampling
6 factor m , the bandpass decimation filter employs complex
7 weights. The filter weights are of course just the bandpass
8 modulated coefficients of a lowpass prototype filter
9 designed to support the bandwidth of the target signal.
10 Samples are collected from the A/D and alternated between
11 the two modulators. Both modulators are identical and use
12 the same predictor filter coefficients. The re-quantized
13 samples are processed by an $m:1$ complex polyphase filter to
14 produce the decimated signal. Several design options are
15 presented once the signal has been filtered and the sample
16 rate lowered. Figure 15 illustrates one possibility. Now
17 that the data rate has been reduced, the low rate signal is
18 easily shifted to baseband with a simple, and area
19 efficient, complex heterodyne. One multiplier and a single
20 digital frequency synthesizer could be time shared to
21 extract one or multiple channels.

22 It is interesting to investigate some of the changes
23 that are required to support the $\Sigma\Delta$ decimator. The center
24 frequency of the prediction filter should be designed to
25 predict samples in the required spectral region in
26 accordance with the output sample rate. For example,
27 consider $m=2$, and the required channel center frequency
28 located at 0.1 Hz, normalized with respect to the input
29 sample rate. The prediction filter should be designed with a
30 center frequency located at 0.2 Hz. In addition, the quality
31 of the prediction should be improved. With respect to the

SUBSTITUTE SPECIFICATION

1 output sample rate, the predictors are required to operate
2 over a wider fractional bandwidth. This implies more filter
3 coefficients in $P(z)$. The increase in complexity of this
4 component should be balanced against the savings that result
5 in the reduced complexity filter stage to confirm that a net
6 savings in logic requirements is produced. To more clearly
7 demonstrate the approach, consider a 2:1 decimator, a
8 channel center frequency at 0.2 Hz and a 60 dB dynamic range
9 requirement.

10 Figure 16(a) shows the double-sided spectrum of the
11 input test signal. The input signal is commutated between $\Sigma\Delta_0$
12 and $\Sigma\Delta_1$ to produce the two low-precision sequences $\hat{x}_0(n)$ and
13 $\hat{x}_1(n)$. The respective spectra of these two signals are shown
14 in Figures 16(b) and 16(c). The complex decimation filter
15 response is defined in Figure 16(d). After filtering, a
16 complex sample stream supported at the low output sample
17 rate is produced. This spectrum is shown in Figure 16(e).
18 Observe that the out-of-band components in the test signal
19 have been rejected by the specified amount and that the in-
20 band data meets the 60 dB dynamic range requirement. For
21 comparison, the signal spectrum resulting from applying the
22 processing stages in the order, re-quantize, filter and
23 decimate is shown in Figure 16(f). The interesting point to
24 note is that while the dual $\Sigma\Delta$ modulator approach satisfies
25 the system performance requirements, its out-of-band
26 performance is not quite as good as the response depicted in
27 Figure 16(f). The stopband performance of the dual modulator
28 architecture has degraded by approximately 6 dB. This can be
29 explained by noting that the shaping noise produced by each
30 modulator is essentially statistically independent. Since
31 there is no coupling between these two components prior to

SUBSTITUTE SPECIFICATION

1 filtering, complete phase cancellation of the modulator
2 noise cannot occur in the polyphase filter.

3 SUMMARY

4 Sigma-delta modulation ($\Sigma\Delta$) technology, together with
5 FPGA based signal processing hardware, is combined to
6 produce creative, high-performance and area-efficient
7 solutions to many signal processing problems.

8 In one embodiment of the invention, a conventional DC
9 canceler is modified to include a re-quantizer in the
10 feedback loop in the form of a $\Sigma\Delta$ modulator. In such
11 embodiments, the modulator can be a very simple 1st order
12 loop that can easily be implemented using an FPGA.

13 In another embodiment, a digital receiver employs a
14 processing chip, such as an FPGA, that includes a $\Sigma\Delta$
15 modulator to requantize oversampled control signals in the
16 digital receiver. This embodiment enables an FPGA to
17 generate analog signals to control low-bandwidth analog
18 functions using minimal additional hardware.

19 Still another embodiment of the invention is a wide-
20 bandwidth sigma-delta loop with a tunable center frequency.
21 In this embodiment, a fixed set of feedback weights from a
22 set of digital integrators defines a base-band filter with a
23 desirable noise transfer function. The filter is tuned to
24 arbitrary frequencies using a simple sub-processing element.
25 The low-pass to band-pass transformation for a sampled data
26 filter is achieved using an all-pass transfer function $G(z)$.
27 In this embodiment, tuning is trivially accomplished by
28 changing a multiplier in the all-pass network. The tuning
29 multipliers can be implemented as full multipliers in FPGA

SUBSTITUTE SPECIFICATION

1 hardware or as dynamically re-configured constant-
2 coefficient multipliers.

3 This summary does not purport to define the invention,
4 which is instead defined by the claims.

5

6 **BRIEF DESCRIPTION OF THE FIGURES**

7 Figure 1 depicts a generic FPGA architecture.

8 Figure 2 depicts a configurable logic block (CLB) of a
9 programmable logic device.

10 Figure 3 shows the CLB matrix used in Xilinx Virtex
11 FPGA's.

12 Figure 4 shows a logic block architecture employed in
13 the Atmel AT40K FPGA.

14 Figure 5 shows a single-loop sigma-delta modulator.

15 Figure 6 shows a linearized discrete time model of a
16 single-loop sigma-delta modulator.

17 Figure 7 depicts the noise transfer function of the
18 modulator of Figure 6.

19 Figure 8 depicts a filter system that includes a sigma-
20 delta modulator.

21 Figure 9 depicts a single loop error feedback sigma-
22 delta modulator.

23 Figure 10 depicts a tunable sigma-delta modulator using
24 a linear filter in the feedback path.

25 Figure 11A is a frequency domain plot of an input test
26 signal for the modulator in Figure 10.

27 Figure 11B is a frequency domain plot of the input test
28 signal of Figure 10.

29 Figure 11C is a frequency domain plot of the frequency
30 response of a 160-tap filter $H(z)$.

31 Figure 11D is a frequency domain plot of a signal
32 resulting from the application of filter $H(z)$ of Figure 11C

SUBSTITUTE SPECIFICATION

1 to re-quantized test signal of Figure 11B.

2 Figure 12 is a frequency domain plot of band-limited
3 white noise process $N_x(\omega)$.

4 Figure 13 depicts an FPGA FIR filter.

5 Figure 14 depicts a look-up table storing 16 scaled co-
6 efficient values for one tap of the FIR filter of Figure 13.

7 Figure 15 depicts a decimation filter that includes two
8 modulators operating in parallel.

9 Figure 16(a) shows the double-sided spectrum of an
10 input test signal for the decimators of Figure 15.

11 Figures 16(b) depicts the spectrum of sequence $\hat{x}_0(n)$ in
12 the decimator of Figure 15.

13 Figures 16(c) depicts the spectrum of sequence $\hat{x}_1(n)$ in
14 the decimator of Figure 15.

15 Figure 16(d) depicts the complex filter response for
16 the decimation filter of Figure 15.

17 Figure 16(e) depicts a complex sample stream produced
18 by the decimation filter of Figure 15.

19 Figure 16(f) depicts a signal spectrum resulting from
20 applying the filter processing stages of Figure 15 in a
21 different order.

22 Figure 17 depicts a fourth-order sigma-delta loop.

23 Figure 18 depicts the time and spectrum obtained by
24 using the loop of Figure 17 with a 4-bit quantizer.

25 Figure 19(a) is a block diagram of a digital filter
26 with the transfer function $G(z)$.

27 Figure 19(b) is a block diagram of a digital filter
28 with the transfer function for $G(z)$ and having a co-
29 efficient multiplier C.

30 Figure 20 depicts a tunable fourth order sigma-delta
31 loop.

SUBSTITUTE SPECIFICATION

1 Figure 21 depicts the time and spectrum obtained by
2 using the tunable loop shown in Figure 20.

3 Figure 22 is a graph depicting the relationship between
4 the coefficient c and the center frequency for the loop of
5 Figure 20.

6 Figure 23 depicts FPGA hardware adapted to implement a
7 loadable constant coefficient multiplier.

8 Figure 24 depicts a simple DC canceler.

9 Figure 25A is a spectral domain representation of a
10 biased signal presented to the DC canceler of Figure 24.

11 Figure 25B is the processed signal spectrum at $yq(n)$ in
12 the DC canceler of Figure 24.

13 Figure 25C depicts the signal spectrum at $yq(n)$ using
14 the quantizer $Q(.)$.

15 Figure 25D demonstrates the operation of the DC
16 canceler of Figure 26 for 8-bit output data.

17 Figure 26 depicts a DC canceler with a re-quantizer
18 embedded in the feedback loop.

19 Figure 27 depicts a digital receiver with feedback
20 paths containing digital signals converted to analog control
21 signals for analog components.

22 Figure 28 depicts a digital receiver employing a sigma-
23 delta modulator to facilitate simplified the feedback
24 circuitry.

25 Figure 29 shows the time responses of the one-bit two
26 loop sigma-delta converter to a slowly varying control
27 signal and the reconstructed signal obtained from the dual-
28 RC filter.

29 Figure 30 shows the spectrum obtained from a 1-bit two-
30 loop modulator and the spectrum obtained from an unbuffered
31 RC-RC filter.

1

2 DETAILED DESCRIPTION

3 To provide a frame of reference for the $\Sigma\Delta$ decimator,
4 consider an implementation that does not pre-process the
5 input data, but just applies it directly to a polyphase
6 decimation filter. A complex filter processing real-valued
7 data consumes double the FPGA resources of a filter with
8 real weights. For $N=160$, 15344 CLBs are required. This
9 Figure is based on a cost of 40 CLBs for each KCM and 8 CLBs
10 for an add-delay component.

11 Now consider the logic accounting for the dual
12 modulator approach. The area cost $\Gamma(\widehat{\text{FIR}})$ for this filter is

13

14 Equation 22:

$$\Gamma(\widehat{\text{FIR}}) = 2\Gamma(\Sigma\Delta) + \Gamma(\widehat{\text{MUL}}) + \Gamma(\text{ACC}_z^{-1})$$

15

16 where $\Gamma\Sigma\Delta$ represents the logic requirements for one $\Sigma\Delta$
17 modulator, and $\Gamma(\widehat{\text{MUL}})$ is the logic needed for a reduced
18 precision multiplier. Using the filter specifications
19 defined earlier, and 18-tap error prediction filters,

20

$$\Gamma(\widehat{\text{FIR}}) = 2 \times 738 + 2 \times ((160 + 159) \times 8) = 6596.$$

22

23 Comparing the area requirements of the two options produces
24 the ratio

25

26 Equation 23:

$$\lambda = \frac{\Gamma(\widehat{\text{FIR}})}{\Gamma_{\text{FIR}}} = 6596/15344 \approx 43\%$$

SUBSTITUTE SPECIFICATION

1

2 So for this example, the re-quantization approach has
3 produced a realization that is significantly more area
4 efficient than a standard tapped-delay line implementation.

5

6 Center Frequency Tuning

7 For both the single-rate and multi-rate $\Sigma\Delta$ based
8 architectures, the center frequency is defined by the
9 coefficients in the predictor filter and the coefficients in
10 the primary filter. The constant coefficient multipliers can
11 be constructed using the FPGA function generators configured
12 as RAM elements. When the system center frequency is to be
13 changed, the system control hardware would update all of the
14 tables to reflect the new channel requirements. If only
15 several channel locations are anticipated, separate
16 configuration bit streams could be stored, and the FPGA(s)
17 re-configured as needed.

18

19 Bandpass $\Sigma\Delta$ s Using Allpass Networks

20 In an earlier section we discussed how to design a
21 predicting filter for the feedback loop of a standard sigma
22 delta modulator. The predicting filter increases the order
23 of the modulator so that the modified structure has
24 additional degrees of freedom relative to a single-delay
25 noise feedback loop. These extra degrees of freedom have
26 been used in two ways, first to broaden the bandwidth of the
27 loop's noise transfer function, and second to tune its
28 center frequency. The tuning process entailed an off line
29 solution of the Normal equations which, while not difficult,
30 does present a small delay and the need for a background
31 processor. We can define a sigma-delta loop with a

SUBSTITUTE SPECIFICATION

1 completely different architecture that offers the same
2 flexibility, namely wider bandwidth and a tunable center
3 frequency that does not require this background task. In
4 this alternate architecture, a fixed set of feedback weights
5 from a set of digital integrators defines a base-band
6 prototype filter with a desirable NTF. The filter is tuned
7 to arbitrary frequencies by attaching to each delay element
8 z^{-1} , a simple sub-processing element that performs a base-
9 band to band-pass transformation of the prototype filter.
10 This processing element tunes the center frequency of its
11 host prototype with a single real and selectable scalar. The
12 structure of a fourth order prototype sigma-delta loop is
13 shown in Figure 17. The time and spectrum obtained by using
14 the loop with a 4-bit quantizer is shown in Figure 18. In
15 this structure the digital integrator poles are located on
16 the unit circle at DC. The local feedback (a_1 and a_2)
17 separates the poles by sliding them along the unit circle,
18 and the global feedback (b_1 , b_2 , b_3 and b_4) places these
19 poles in the feedback path of the quantizer so they become
20 noise transfer function zeros. These zeros are positioned to
21 form an equal-ripple stop band for the NTF. The coefficients
22 selected to match the NTF pole-zero locations to an elliptic
23 high pass filter. The single sided bandwidth of this fourth
24 order loop is approximately 4% of the input sample rate.

25 The low-pass to band-pass transformation for a sampled
26 data filter is achieved by substituting an all-pass transfer
27 function $G(z)$ for the all-pass transfer function z^{-1} . This
28 transformation is shown in equation 24.

29

SUBSTITUTE SPECIFICATION

1 Equation 24:

$$z^{-1} \longrightarrow -z^{-1} \left(\frac{1-cz}{z-c} \right)$$

2

3

4 A block diagram of a digital filter with the transfer
5 function for $G(z)$ is shown in Figure 19. Examining the left
6 hand block diagram, we find the transfer function from $x(n)$
7 to $y(n)$ is the all-pass network $-(1-cz)/(z-c)$, while the
8 transfer function from $x(n)$ to $v(n)$ is $-(1/z)(1-cz)/(z-c)$.
9 When we absorb the external negative sign change in the
10 internal adders of the filter we obtain the simple right-
11 hand side version of the desired transfer function $G(z)$.

12 After the block diagram substitution has been made, we
13 obtain Figure 20, the tunable version of the low-pass
14 prototype. The basic structure of the prototype remains the
15 same when we replace the delay with the tunable all-pass
16 network. The order of the filter is doubled by the
17 substitution since each delay is replaced by a second order
18 sub-filter. Tuning is trivially accomplished by changing the
19 c multiplier of the all-pass network. The tuned version of
20 the system reverts back to the prototype response if we set
21 c to 1.

22 Figure 21 presents the time and spectrum obtained by
23 using the tunable loop with a 4-bit quantizer shown in
24 Figure 20. The single sided bandwidth of the prototype
25 filter is distributed to the positive and negative spectral
26 bands of the tuned filter. Thus the two-sided bandwidth of
27 each spectral band is approximately 4% of the input sample
28 rate.

SUBSTITUTE SPECIFICATION

1 We now estimate the computational workload required to
2 operate the prototype and tunable filter. The prototype
3 filter has six coefficients to form the 4-poles and the 4-
4 zeros of the transfer function. The two a_k $k=0,1$ coefficients
5 determine the four zero locations. These are small
6 coefficients and can be set to simple binary scalers. The
7 values computed for this filter for a_1 and a_2 were 0.0594 and
8 0.0110. These can be approximated by $1/16$ and $1/128$, which
9 lead to no significant shift of the spectral zeros in the
10 NTF. These simple multiplications are virtually free in the
11 FPGA hardware since they are implemented with suitable
12 wiring. The four coefficients b_k $k=0,...,3$ are 1.000, 0.6311,
13 0.1916, and 0.0283 respectively were replaced with
14 coefficients containing one or two binary symbols to obtain
15 values 1.000 , $1/2+1/8$ (0.625), $1/8+1/16$ (0.1875) and $1/32$
16 (0.03125). When the sigma-delta loop ran with these
17 coefficients there was no discernable change in bandwidth or
18 attenuation level of the loop. The loop operates equally as
19 well in the tuning mode and the non-tuning mode with the
20 approximate coefficients listed above. Thus the only real
21 multiplies in the tunable sigma-delta loop are the c
22 coefficients of the all-pass networks. These networks are
23 unconditionally stable and always exhibit all-pass behavior
24 even in the presence of finite arithmetic and finite
25 coefficients. This is because the same coefficient forms the
26 numerator and the denominator. Errors in approximating the
27 coefficients for c simply result in a frequency shift of the
28 filter's tuned center. The c coefficient is determined from
29 the cosine of the center frequency (in radians/sample). The
30 curve for this relationship is shown in Figure 22. Also
31 shown is an error due to approximating c by $c+\delta c$. The

SUBSTITUTE SPECIFICATION

1 question is, what is the change in center frequency θ , from
2 $\theta + \delta\theta$ due to the approximation of c ? We can see that the slope
3 at the operating point on the cosine curve is $-\sin\theta$ so that
4 $\delta c / \delta\theta \approx -\sin(\theta)$ so that $\delta c \approx -\delta\theta \sin(\theta)$ is the required
5 precision to maintain a specified error. We note that tuning
6 sensitivity is most severe for small frequencies where $\sin(\theta)$
7 is near zero. The tolerance term, $\delta\theta \sin(\theta)$, is quadratic for
8 small frequencies, but the lowest frequency that can be
9 tuned by the loop is half the NTF pass-band bandwidth. For
10 the fourth order system described here, this bandwidth is 4%
11 of the sample rate, so the half-bandwidth angle is 2%, or
12 0.126 radians. To assure that the frequency to which the
13 loop is tuned has an error smaller than 1% of center
14 frequency, $\delta c < \delta\theta \sin(\theta) \Rightarrow \delta c < (0.126/100)(0.126) = 0.0002$,
15 which corresponds to a 14 bit coefficient. An error of less
16 than 10% center frequency can be achieved with 10 bit
17 coefficients.

18 The tuning multipliers could be implemented as full
19 multipliers in the FPGA hardware or as dynamically re-
20 configured KCMs, or KDCM, as shown in Figure 23. The later
21 approach conserves FPGA resources at the expense of
22 introducing a start-up penalty each time the center
23 frequency is changed. The start-up period is the
24 initialization time of the KCM LUT. When a new center
25 frequency is desired, the tuning constant is presented to
26 the k input of the KDCM and the load signal LD is asserted.
27 This starts the initialization engine, which requires 16
28 clock cycles to initialize 16 locations in the multiplier
29 LUT. The initialization engine relies on the automatic shift
30 mode of the Virtex LUTs. In this mode of operation a LUT's
31 register contents are passed from one cell to the next cell

SUBSTITUTE SPECIFICATION

1 on each clock tick. This avoids the requirement for a
2 separate address generator and multiplexor in the
3 initialization hardware. Observe from Figure 23 that the
4 initialization engine only introduces a small amount of
5 additional hardware over that of a static KCM.

6 There is approximately a factor of 4 difference in the
7 area of a KDCM and full multiplier.

8
9 $\Sigma\Delta$ DC Canceler

10 Unwanted DC components can be introduced into a DSP
11 datapath at several places. It may be presented to the
12 system via an un-trimmed offset in the analog-to-digital
13 conversion pre-processing circuit, or may be attributed to
14 bias in the A/D converter itself. Even if the sampled input
15 signal has a zero mean, DC content can be introduced through
16 arithmetic truncation processes in the fixed-point datapath.
17 For example, in a multi-stage multi-rate filter, the
18 intermediate filter output samples may be quantized between
19 stages in order to compensate for the filter processing gain
20 and thereby keep the word-length requirements manageable.
21 The introduced DC bias can impact the dynamic range
22 performance of a system and potentially increase the error
23 rate in a digital receiver application.

24 In a fixed-point datapath, the bias can cause
25 unnecessary saturation events that would not occur if the DC
26 was not present in the system.

27 In a digital communication receiver employing M-ary QAM
28 modulation, the DC bias can interfere with the symbol
29 decision process, so causing incorrect decoding and
30 therefore increasing the bit error rate.

SUBSTITUTE SPECIFICATION

1 In some cases the introduced bias can be ignored and is
2 of no concern. However, for other applications it is
3 desirable to remove the DC component.

4 One solution to removing the unwanted DC level is to
5 employ a DC canceler.

6 A simple canceler is shown in Figure 24. It is easy to
7 show that the transfer function of the network is

8

9 Equation 25:

$$H(z) = \frac{z-1}{z-(1-z)}$$

10

11

12 The cancellation is due to the transfer function zero at 0
13 Hz. The pole at $1-\mu$ controls the system bandwidth, and hence
14 the system transient response. The location of the zero at
15 $z=1$ removes the DC component in the signal, but there are
16 some problems with a practical implementation of this
17 circuit.

18 Figure 25A is a spectral domain representation of a
19 biased signal presented to the DC canceler. Figure 25B is
20 the processed signal spectrum at $y_q(n)$ in Figure 24. We
21 observe that the DC content in the input signal has been
22 completely removed. However, in the process of running the
23 canceling loop the network processing gain has caused a
24 dynamic range expansion. So although the sample stream $y_q(n)$
25 is a zero mean process, it requires a larger number of bits
26 to represent each sample than is desirable. The only option
27 with the circuit is to re-quantize $y_q(n)$ to produce $y(n)$
28 using the quantizer $Q(\cdot)$. The effect of this operation is
29 shown in Figure 25C, which demonstrates, not surprisingly,

SUBSTITUTE SPECIFICATION

1 that after an 8-bit quantizer, the signal now has a DC
2 component and we are almost back to where we started. How
3 can the canceler be re-organized to avoid this
4 implementation pitfall? One option is to embed the re-
5 quantizer in the feedback loop in the form of a $\Sigma\Delta$ modulator
6 as shown in Figure 26. The modulator can be a very simple
7 1st order loop such as the error feedback $\Sigma\Delta$ modulator shown
8 in Figure 9. Figure 25D demonstrates the operation of the
9 circuit for 8-bit output data. Observe from the Figure that
10 the DC has been removed from the signal while employing the
11 same 8-bit output sample precision that was used in Figure
12 24. The simple $\Sigma\Delta$ M employed in the canceler is easily
13 implemented in an FPGA.

14

15 **Simplify Digital Receiver Control Loops Using $\Sigma\Delta$ Modulators**

16 In earlier sections we recognized that when a sampled
17 data input signal has a bandwidth that is a small fraction
18 of its sample rate the sample components from this
19 restricted bandwidth are highly correlated. We took
20 advantage of that correlation to use a digital sigma-delta
21 modulator to requantize the signal to a reduced number of
22 bits. The sigma-delta modulator encodes the input signal
23 with a reduced number of bits while preserving full input
24 precision over the signal bandwidth by placing the increased
25 noise due to requantization in out-of-band spectral
26 positions that are already scheduled to be rejected by
27 subsequent DSP processing. The purpose of this
28 requantization is to allow the subsequent DSP processing to
29 be performed with reduced arithmetic resource requirements
30 since the desired data is now represented by a smaller
31 number of bits.

SUBSTITUTE SPECIFICATION

1 A similar remodulation of data samples can be employed
2 for signals generated within a DSP process when the
3 bandwidth of the signals are small compared to the sample
4 rate of the process. A common example of this circumstance
5 is the generation of control signals used in feedback paths
6 of a digital receiver. These control signals include a gain
7 control signal for a voltage controlled amplifier in an
8 automatic gain control (AGC) loop and VCO (voltage
9 controlled oscillator) control signals in carrier recovery
10 and timing recovery loops. A block diagram of a receiver
11 with these specific controls signals is shown in Figure 27.
12 The control signals are generated from processes operating
13 at a sample rate appropriate to the input signal bandwidth.
14 The bandwidth of control loops in a receiver are usually a
15 very small fraction of the signal bandwidth, which means
16 that the control signals are very heavily oversampled. As a
17 typical example, in a cable TV modem, the input bandwidth is
18 6 MHz, the processing sample rate is 20 MHz, and the loop
19 bandwidth may be 50 kHz. For this example, the ratio of
20 sample rate to bandwidth is 400-to-1.

21 As seen in Figure 27, the process of delivering these
22 oversampled control signals to their respective control
23 points entails the transfer of 16 bit words to external
24 control registers, requiring appropriate busses, addressing,
25 and enable lines as well as the operation of 16-bit digital-
26 to-analog converters (DACs).

27 We can use a sigma-delta modulator to requantize the
28 16-bit oversampled control signals in the digital receiver
29 prior to passing them out of the processing chip. The sigma-
30 delta can preserve the required dynamic range over the
31 signal's restricted bandwidth with a one-bit output. As
32 suggested in Figure 28, the transfer of a single bit to

SUBSTITUTE SPECIFICATION

1 control the analog components is a significantly less
2 difficult task than the original. We no longer require
3 registers to accept the transfer, the busses to deliver the
4 bits, or the DAC to convert the digital data to the analog
5 levels the data represents. All that is needed a simple
6 filter (and likely an analog amplifier to satisfy drive
7 level and offset requirements). Experience shows that a 1-
8 bit, one-loop sigma-delta modulator could achieve 80 dB
9 dynamic range and requires a single RC filter to reconstruct
10 the analog signal. A two-loop sigma-delta modulator is
11 required to achieve 16-bit precision for which a double RC
12 filter is required to reconstruct the analog output signal.
13 Figure 29 shows the time response of the one-bit two loop
14 sigma-delta converter to a slowly varying control signal and
15 the reconstructed signal obtained from the dual-RC filter.
16 Figure 30 shows the spectrum obtained from a 1-bit two-loop
17 modulator and the spectrum obtained from an unbuffered RC-RC
18 filter.

19 This example has shown how with minimal additional
20 hardware, an FPGA can generate analog control signals to
21 control low-bandwidth analog functions in a system.
22 An observation worthy of note, is that the audio engineering
23 community has recognized the advantage offered by this
24 option of requantizing a 16-bit oversampled data stream to
25 1-bit data stream. In that community, the output signal is
26 intentionally upsampled by a factor of 64 and then
27 requantized to 1-bit in a process called a MASH converter.
28 Nearly all CD players use the MASH converter to deliver
29 analog audio signals.

30

SUBSTITUTE SPECIFICATION1 What Have We Gained?

2 What has been achieved by expressing our signal
3 processing problems in terms of $\Sigma\Delta$ techniques?

4 The paper has demonstrated some $\Sigma\Delta$ techniques for the
5 compact implementation of certain types of filter and
6 control applications using FPGAs. This optimization can be
7 used in several ways to bring economic benefits to a
8 commercial design. By exploiting $\Sigma\Delta$ filter processes, a
9 given processing load may be realizable in a lower-density,
10 and hence less expensive, FPGA than is possible without
11 access to these techniques. An alternative would be to
12 perform more processing using the same hardware. For
13 example, processing multiple channels in a communication
14 system.

15 In addition to FPGA area trade-offs, the $\Sigma\Delta$ methods
16 can result in reduced power consumption in a design. Power P
17 may be expressed as

18

19 Equation 26:

$$P = CV^2 f_{clk}$$

20

21 where C is capacitance, V is voltage and f is the system
22 clock frequency. By reducing the silicon area requirements
23 of a filter, we can simultaneously reduce the power
24 consumption of the design. For the examples considered
25 earlier, logic resource savings of greater than 50% were
26 demonstrated. The savings is proportional to increased
27 efficiency in the system power budget, and this of course is
28 very important for mobile applications.

29 The $\Sigma\Delta$ AGC, timing and carrier recovery control loop
30 designs are also important examples in a industrial context.

SUBSTITUTE SPECIFICATION

1 The examples illustrated how the component count in a mixed
2 analog/digital system can be reduced. In fact, not only is
3 the component count reduced, but printed circuit board area
4 is minimized. This results in more reliable and physically
5 smaller implementations. The reduced component count also
6 results in reduced power consumption. In addition, since the
7 control loops no longer require wide output buses from the
8 FPGA to multi-bit DACs that generate analog control
9 voltages, power consumption is decreased because fewer FPGA
10 I/O pads are being driven.

11

12 References

13 The subject matter of this application is excerpted
14 from an article entitled "FPGA Signal Processing Using
15 Sigma-Delta Modulation," C. H. Dick and F. J. Harris, IEEE
16 Signal Processing Magazine (January 2000), which is
17 incorporated herein by reference. The following documents
18 may also be of interest, and are also incorporated by
19 reference.

20 [1] Atmel, AT40K/05/10/20/40 Data Sheet, 1999.

21 [2] J. A. C. Bingham, The Theory and Practice of Modem
22 Design, John Wiley & Sons, New York, 1998.

23 [3] J. C. Candy and G. C. Temes, "Oversampling Methods for
24 A/D and D/A Conversion" in Oversampling Delta Sigma
25 Data Converters - Theory Design and Simulation, IEEE
26 Press, New York, 1992.

27 [4] M. E. Frerking, Digital Signal Processing in
28 Communication Systems, Van Nostrand Reinhold, New York,
29 1994.

30 [5] S. Haykin, "Modern Filters", Macmillan Publishing Co.,
31 New York, 1990.

SUBSTITUTE SPECIFICATION

- 1 [6] H. Meyr, M. Moeneclaey and S. A. Fechtal, Digital
2 Communication Receivers, John Wiley & Sons Inc., New
3 York, 1998.
- 4 [7] S. R. Norsworthy, R. Schreier and G. C. Temes, \Delta-
5 Sigma Data Converters", IEEE Press, Piscataway, NJ,
6 1997.
- 7 [8] D. A. Patterson and J. L. Hennessy, Computer
8 Architecture: A Quantitative Approach, Morgan Kaufmann
9 Publishers Inc., California, 1990.
- 10 [9] A. Peled and B. Liu, "A New Hardware Realization of
11 Digital Filters", IEEE Trans. on Acoust., Speech,
12 Signal Processing, vol. 22, pp. 456-462, Dec. 1974.
- 13 [10] J. G. Proakis, and D. G. Manolakis, Digital Signal
14 Processing Principles, Algorithms and Applications
15 Second Edition, Maxwell Macmillan International, New
16 York, 1992.
- 17 [11] S. A. White, "Applications of Distributed Arithmetic to
18 Digital Signal Processing", IEEE ASSP Magazine, Vol.
19 6(3), pp. 4-19, July 1989.
- 20 [12] Xilinx Inc., The Programmable Logic Data Book, 1999.